HPC(期中大猜題)

Ch1:

刀鋒伺服器(blade server):

有一個完整的基座，以統一集中的方式，提供電源、風扇散熱，網路等功能。而基座上可插置多張單板電腦，因狀似刀片(blade)，因此稱刀鋒伺服器，而基座稱刀風基座。

What is high performance computing?

- ✧ Functional requirement
- ✧ Non-functional requirement
  - Speed,quality,space,user friendliness,

Three ways to improve speed

1. Work harder
2. Work smarter
3. Get help

Why PC Cluster?

- ✧ PC 及 networking 的速度越來越快
- ✧ 大型電腦中心工作等候時間過長
- ✧ 受限經費，無法採購大型主機

What is Cluster Computing?

- ✧ A collection of interconnected computers working (computing) together as a single system.
- ✧ The nodes of a cluster can exist in a single cabinet or be physically separated and connected via a LAN

Cluster Usage Modes

- ✧ NOW(network of workstation)
- ✧ PMMPP (poor man 的 massive parallel processor)

Three schemes are used to share cluster nodes

- ✧ Dedicated mode (dedicated:專用)
- ✧ Space sharing
- ✧ Time sharing

單工 vs 多工
Turnaround time(平均完成時間) =>單工 better
Throughput(單位時間產出)=>多工 better

Types of Parallel Jobs
- Rigid (依程式去指定固定數量的處裡器)
- Moldable(執行前可任意更動處理器數量，執行期間不行)
- Evolving(由系統調控處理器數量)
- Malleable (由程式去調控處理器數量)

HPC as a Service
- Ease of use
- Dynamic allocation
- Elastic and fast resource provisioning

Three speedup models:
- Linear speedup model
- Amdahl’s law
- Downey’s speedup model

Two approaches:
- Moldable job scheduling without job execution time Information
    Run as many jobs as possible simultaneously
    One job after another sequentially
- Moldable job scheduling with job execution time Information
  raising resource utilization rate to reduce average turnaround time for all jobs.

Future challenge:
- Budget parameters
- Time to completion
- Monitor and steer worklow
- Obtain Interactive results

Ch2:
矩鎮相乘的 4 種方法:
1.Loop-i-j-k(基本款)

```
Int I,j,k,F;
For(i=0;i<n;i++)
```

```
{
        For(j=0;j<n;j++)
        {
                F = 0;
                For(k=0;k<n;k++)
                {
                        F*= A[i][k]*B[k][j];
                }
                C[i][j] = F;
        }
}
```

2.Loop-reordering(將 loop-i-j-k 的內層互換並做微調)

```
For(i=0;i<m;i++)
  For(k=0;k<n;k++)
  {
        F = A[i][k];
        For(j=0;j<p;j++)
        {
                C[i][j] += f*B[k][j];
        }
  }
```

3　matrix-column(loop-i-j-k 外層互調並微調)

matrix-vecter multiplication

4　block matrix (loop-i-j-k 外加兩層，控制 block 象限的移動)

```
for(k=0;k<n;k+=p)
        {
                for(j=0;j<n;j+=p)
                {
                        for(i=0;i<n;i++)
                        {
                                for(jj=j;jj<min(j+p,n);jj++)
                                {
                                        for(kk=k;kk<min(k+p,n);kk++)
                                                C[i][jj]+=A[i][kk]*B[kk][jj];
                                }
```

```
            }
        }
    }
```

Performance : 5>4>3>2>1

Ch4:

$$\mathrm{Paint}(i,j)=\begin{cases}\phi, & \text{if } i=0\\ \mathrm{Paint}'(i,j), & \text{otherwise}\end{cases}$$

$$\mathrm{Paint}'(i,j)=\begin{cases}\mathrm{Paint0}(i,j), & \text{if Fix0}(i,j)\text{ and not Fix1}(i,j)\\ \mathrm{Paint1}(i,j), & \text{if not Fix0}(i,j)\text{ and Fix1}(i,j)\\ \mathrm{Merge}(\mathrm{Paint0}(i,j),\mathrm{Paint1}(i,j)), & \text{otherwise}\end{cases}$$

$$\mathrm{Paint0}(i,j)=\mathrm{Paint}(i-1,j)0$$

$$\mathrm{Paint1}(i,j)=\mathrm{Paint}(i-d_j-1,j-1)\sigma(d_j)$$

$$\mathrm{Merge}(s_1s_2\dots s_i,t_1t_2\dots t_i)=m_1m_2\dots m_i,\qquad \text{where } m_k=\mathrm{Merge}C(s_k,t_k)$$

$$\mathrm{Merge}C(s_k,t_k)=\begin{cases}0, & \text{if } s_k=t_k=0\\ 1, & \text{if } s_k=t_k=1\\ u, & \text{otherwise.}\end{cases}$$

Paint(i,j) call Paint’ (i,j)

Paint’ s(i,j) call paint0 ,paint1

Paint0 > Fix0 OK

Paint1 > Fix1 OK

Merge(paint0,paint1) > Fix0 OK ,Fix1 OK

$$\mathrm{Fix}(i,j)=\begin{cases}\text{true}, & \text{if } i=0 \text{ and } j=0\\ \text{false}, & \text{if } i=0 \text{ and } j\geq 1\\ \mathrm{Fix0}(i,j)\vee\mathrm{Fix1}(i,j), & \text{otherwise}\end{cases}$$

$$\mathrm{Fix0}(i,j)=\begin{cases}\mathrm{Fix}(i-1,j), & \text{if } s_i\in\{0,u\}\\ \text{false}, & \text{otherwise}\end{cases}$$

$$\mathrm{Fix1}(i,j)=\begin{cases}\mathrm{Fix}(i-d_j-1,j-1), & \text{if } j\geq 1, i\geq d_j+1, \text{ and } s_{i-d_j}\dots s_i \text{ matches } \sigma(d_j)\\ \text{false}, & \text{otherwise.}\end{cases}$$

procedure FP1(G)

1. *Initialize* $G_{p,0}, G_{p,1}$ *for all unpainted pixel* p
2. **repeat**
3. PROPAGATE(G)
4. **if** ($\mathrm{status}(G)$ *is* CONFLICT *or* SOLVED) **then return**
5. UPDATEONALLG(G)
6. **for** (*each unpainted pixel* p *in* G) **do**
7. PROBE(p)
8. **if** ($\mathrm{status}(G)$ *is* CONFLICT *or* SOLVED) **then return**
9. **if** ($\mathrm{status}(G)$ *is* PAINTED) **then break**
10. **end for**
11. **until** $\Pi(G) = \emptyset$

end procedure

Procedure PROBE(p)

1. PROBEG($p, 0$); // guess $p = 0$ and probe $G_{p,0}$
2. PROBEG($p, 1$); // guess $p = 1$ and probe $G_{p,1}$
3. **if** (*both* $\mathrm{status}(G_{p,0})$ *and* $\mathrm{status}(G_{p,1})$ *are* CONFLICT) **then** $\mathrm{status}(G) \leftarrow$ CONFLICT; **return**
4. **if** ($\mathrm{status}(G_{p,0})$ *is* CONFLICT) **then**
5. *Let* Π *be the set of newly painted pixels in* $G_{p,1}$ *with respect to* G
6. **else if** ($\mathrm{status}(G_{p,1})$ *is* CONFLICT) **then**
7. *Let* Π *be the set of newly painted pixels in* $G_{p,0}$ *with respect to* G
8. **else**
9. *Let* Π *be the set of pixels with the same value 0 or 1 in both* $G_{p,0}$ *and* $G_{p,1}$ *with respect to* G
10. **end if**
11. **if** ($\Pi \neq \emptyset$) **then** UPDATEONALLG(Π); $\mathrm{status}(G) \leftarrow$ PAINTED
12. **else** $\mathrm{status}(G) \leftarrow$ INCOMPLETE

end procedure